Technocolabs Data Science Internship

DA Project Report

TITLE: Lending Club's Loan Repayment Prediction

AIM:

The main purpose of this project is to develop machine learning models (Logistic Regression and Neural Networks) on Lending Club's data and deploy them for wider general-purpose usage.

ABSTRACT:

Traditionally, loan-level risk is measured as credit risk—the probability of default to measure the expected loss. Using machine learning techniques, we modeled credit risk and expected payoff maximization on the ROC, to help Lending Club optimize their risk. The models used here are Logistic Regression and Neural Networks.
This project aims to analyze the loans that were paid off in full or charged off.

INTRODUCTION:

Lending Club's dataset contains extensive information of their customer's loan status and many other metrics which help in the analysis of its data. The goal here is to build a model using this data and use that model to accurately predict whether the loans were either fully paid or charged off.

OVERVIEW:

1. Examine and Clean the Dataset
2. Visualize the dataset for gaining clarity
3. Build and Train the model
4. Visualize the Model's end Report

DATASET:

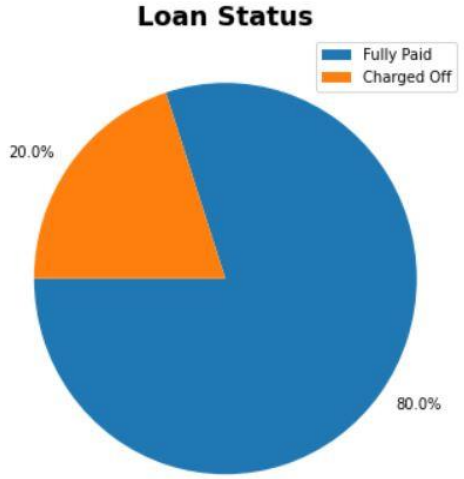
The Lending Club Dataset Contains 2260701 rows by default, which detail the number of customers and it has 151 columns which contain all details with respect to their lending club loan data requirements.

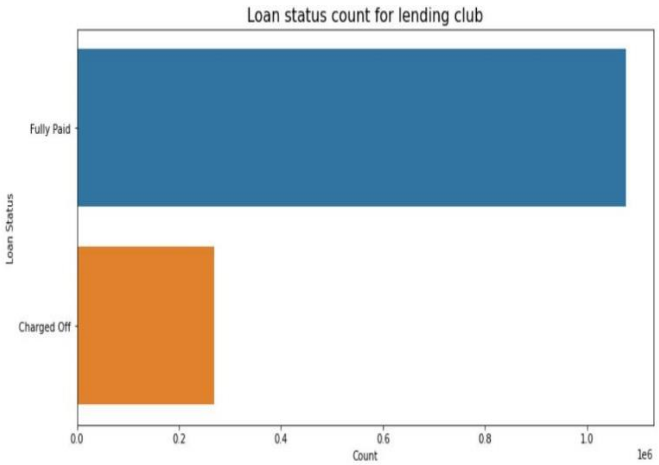
1) EXAMINE AND CLEAN THE DATASET:

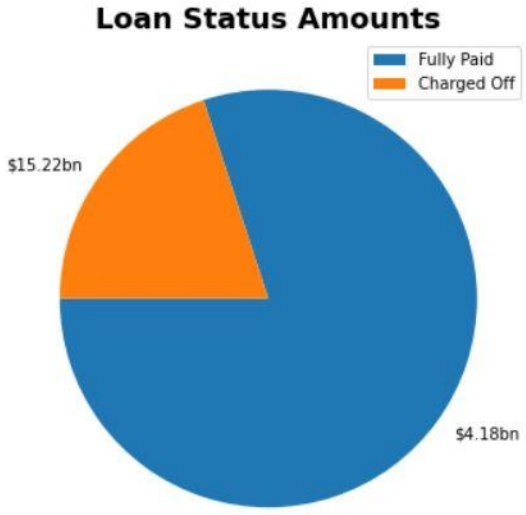
- The dataset is readily available from the bank's own website and via Kaggle.com as well. The data is in the form of .csv(comma-separated values) file. These files need to be loaded in the folder which contains the ipython notebook which will be used for analysis and model building.
- The dataset is then analysed for any discrepancies(mostly null values) and the percentage of null values in every column must be observed. If null values in any column exceeds 70%, then that column needs to be dropped to maintain accuracy.
- Then to analyze the loan status of the dataset, we label the values "Charged Off" as 0 and "Fully Paid" as 1 for proper classification

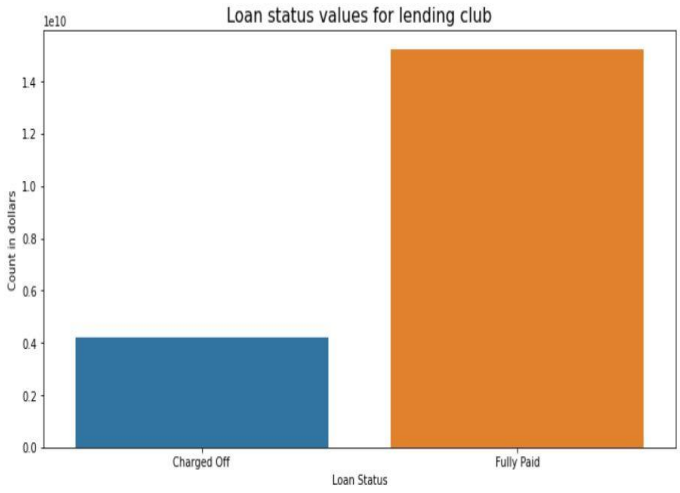
2) VISUALIZING THE DATASET:

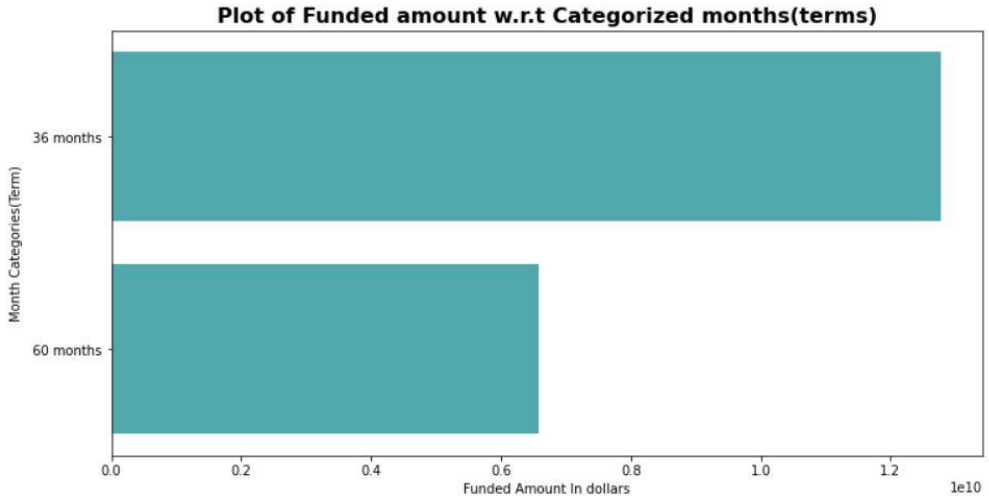
- Here we visualize the dataset parameters using matplotlib and seaborn modules
- The charts produced are:

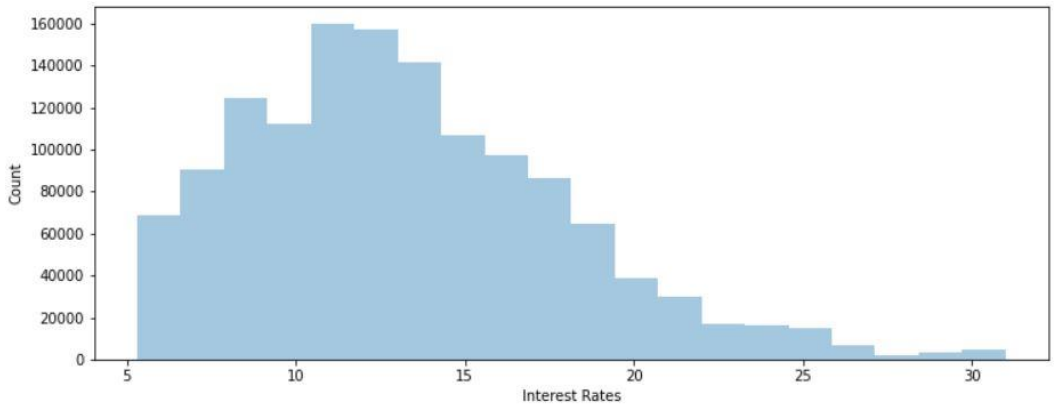


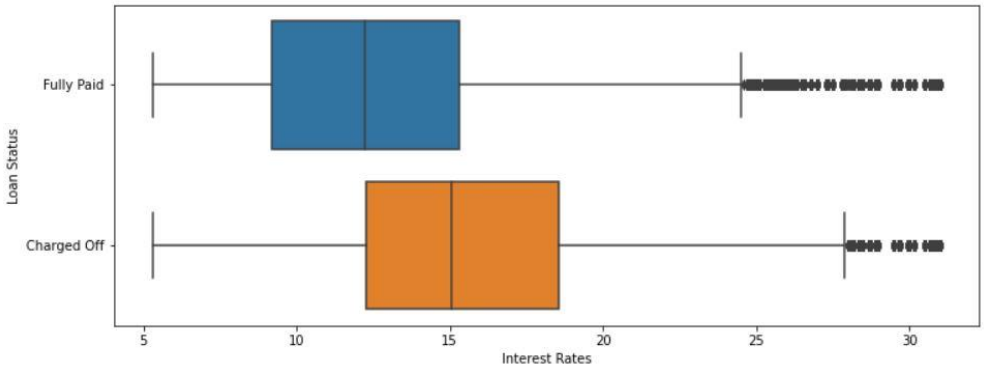


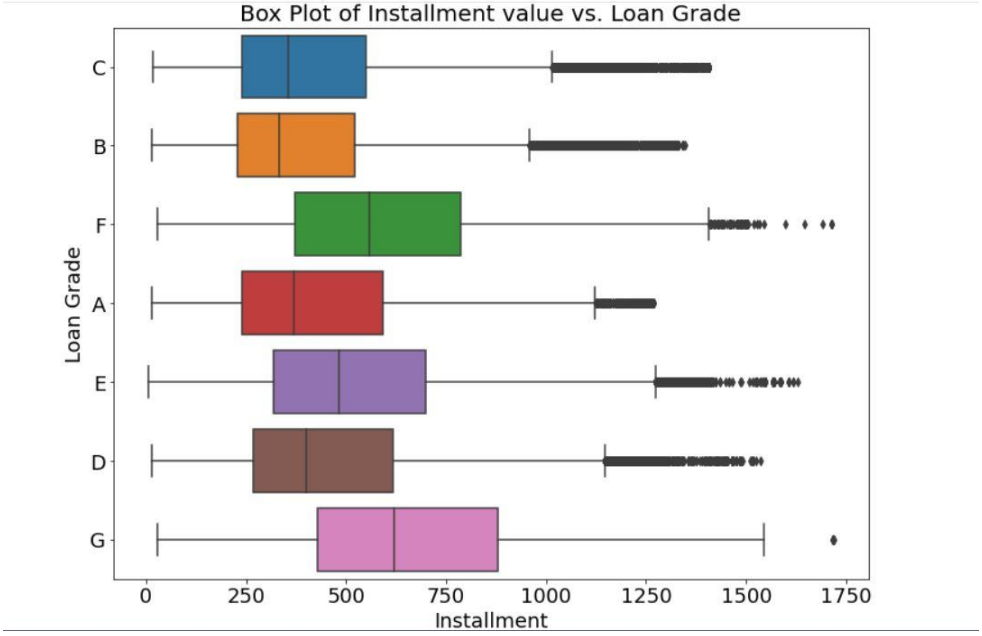





Plot of Funded amount w.r.t Categorized months(terms)
36 months
60 months
Month Categories(Term)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e10
Funded Amount In dollars

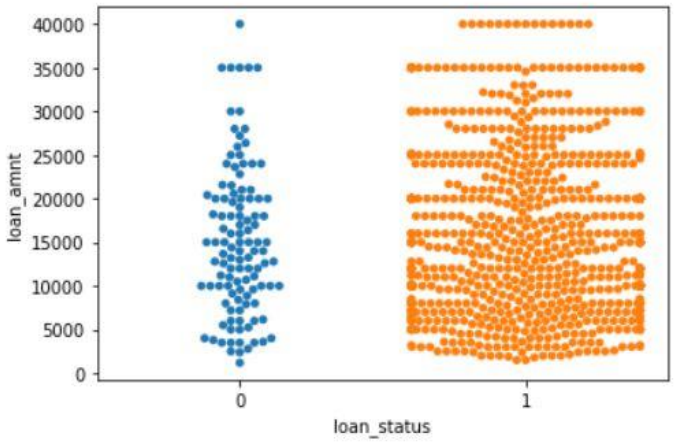

160000
140000
120000
100000
80000
60000
40000
20000
0
Count
5
10
15
20
25
30
Interest Rates

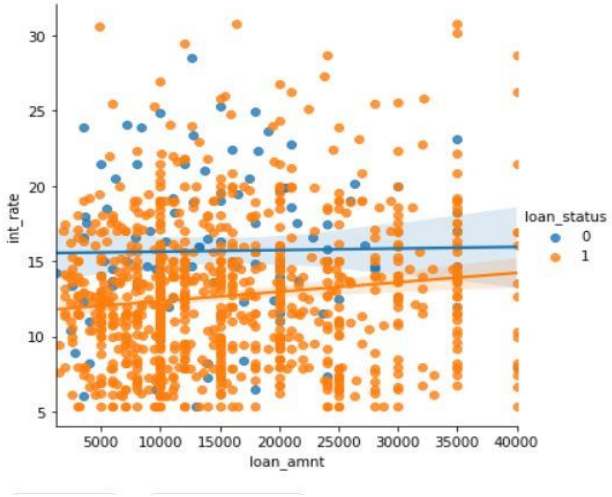

Fully Paid
Charged Off
Loan Status
5
10
15
20
25
30
Interest Rates



The below graphs are:

- Swarmplot of loan amount v/s loan status
- Lmplot of interest rate v/s loan amount



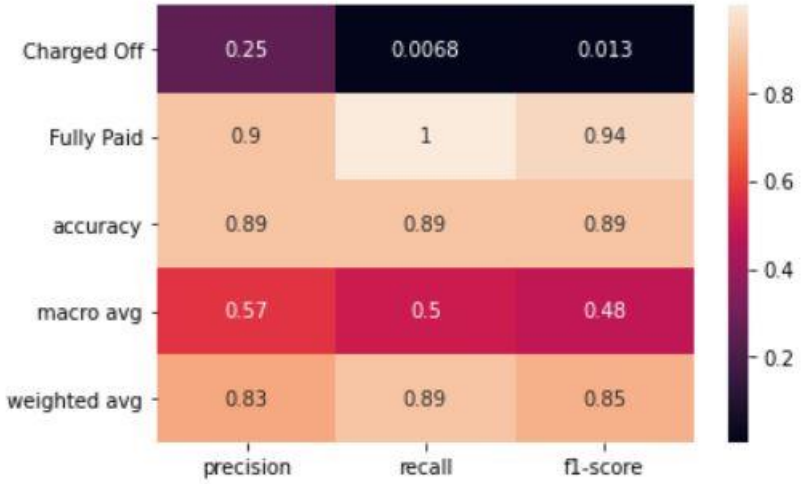


3) BUILDING AND TRAINING THE MODEL:

- The logistic regression model needs to built using numeric data only. Hence the first step is to encode all the columns with non-numeric data types(they are mostly object data type) and assign numeric labels. This is done using One hot encoding.
- Next we assign the target values(y) to be the labelled column of loan status, wherein charged off is labelled as 0 and fully paid is labelled as 1. The feature matrix(X) consists of all the encoded columns which will be used for training the model.
- In the next step we split the data into training and testing sets using the train_test_split function.
- After obtaining the train and test sets we fit the data into the logistic regression model and let it train. After the training ends, we use the model to test it on the test set and obtain predicted values.

4) VISUALIZE THE MODEL'S END REPORT:

- To obtain the accuracy and other metrics of the model, we print out its classification report and subsequently plot a heatmap for easily visualizing it.
- The heatmap is as follows:



- Here as we can see, the model is approximately 89% accurate on unseen data.

DEPLOYMENT OF THE TRAINED AND TESTED MODEL:

- Deployment can be carried out by both flask and streamlit applications. Here we used streamlit.
- In Streamlit, first we saved our model using in a pickle file format(.pkl) and loaded it into a new python file. There we specified html parameters for the look of the end application and created created a function which used this model to display the predictions. The predictions would be displayed as either "Charged Off" or "Fully Paid".
- Then we finally deployed our streamlit model using Heroku for open world use.